Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Mestrado em Engenharia Informática e de Computadores
Semestre Inverno 09/10

Integração de Redes e Serviços

António Borga nº 19710

Introdução

O objectivo deste trabalho pretende ser o desenvolvimento de um sistema computacional que forneça serviços de rede, tendo como objectivo a familiarização com a instalação, configuração e administração de um sistema operativo open-source, nomeadamente como plataforma de suporte a serviços de rede, bem como a instalação, configuração e teste desses serviços. O relatório que se segue descreve essa implementação.

Instalação e configuração do Gentoo

O sistema operativo escolhido foi o Gentoo, e optou-se por instalar uma imagem sobre uma máquina virtual usando o virtualbox para o efeito.

Para a instalação do SO seguiu-se o manual de instalação disponível na página do Gentoo:

http://www.gentoo.org/doc/en/handbook/handbook-x86.xml

No decurso do processo de instalação diversos aspectos foram personalizados:

Sistema de ficheiros projectado:

mount	Device	file system	Dimensão
/boot	/dev/hda1	Ext 2	5M
swap	/dev/hda2	Swap	128M
/	/dev/hda3	Ext 3	6G

Para a configuração do sistema de ficheiros utiliza-se as ferramentas *fdisk*, *mke2fs* e *mkswap* e swapon para o device de swap.

Por forma a trabalhar com o teclado Português evocamos o comando:

loadkeys pt-latin1

Esta definição pode ser tornada persistente editando o ficheiro */etc/conf.d/keymaps* e alterando a flag *KEYMAP="pt-latin1"*

Para a instalações de pacotes definimos o mirror onde queremos ir buscar software, neste caso o mais perto possível:

http://mirrors.ipv4.net.ipl.pt/gentoo

A distribuição pretendida foi a mais recente, disponível:

/snapshots/portage-latest.tar.bz2

Para a compilação dos pacotes ajustamos a variável: *USE="-X -ldap"* no ficheiro */etc/make.conf*

Ao nível de definições de rede atribuímos duas interfaces de *LAN: Eth0* e *Eth1*, com a seguinte configuração em */etc/conf.d/net*

 # eth0
config_eth0=("dhcp")

 # eth1
config_eth1=("10.62.74.220/28")

Na Eth1 definimos a LAN atribuida *10.62.74.208/28*

Criamos dois links em */etc/init.d/*, *net.eth0* e *net.eth1* a partir do script *net.lo*, de forma a permitir a gestão das duas interfaces:

ln –s net.lo net.eth0

Para que as interfaces sejam activadas na inicialização do sistema utilizamos o comando:

rc-update add net.eth0

Este comando permite adicionar (*add*), remover (*del*) e listar (*show*) os serviços que se pretendem iniciar no arranque do sistema.

A manipulação de drivers é efectuada com um conjunto de comandos:

// listar drivers
lsmod

// adicionar driver
modprobe pcnet32

// remover driver
rmmod pcnet32

Na compilação do kernel optou-se inicialmente pelo script disponibilizado pela distribuição gentoo, o *genkernel*, mais tarde verificou-se que este adicionava muito software que não era necessário pelo que fica aqui a indicação de uma compilação mais versátil, a manual:

Na directoria /usr/src/linux utilizamos a ferramenta de gestão de software, módulos, etc, que queremos na nossa configuração:

// menu
make menuconfig

// compilação de módulos
make modules_install

// compilação do kernel
make install

Nesta fase chegamos á configuração do boot loader *grub*, nomeadamente a imagem compilada do kernel que se pretende carregar.

vi /boot/grub/grub.conf

kernel /boot/vmlinuz-2.6.30-gentoo-r8 root=/dev/ram0 real_root=/dev/hda3
initrd /boot/initramfs-genkernel-x86-2.6.30-gentoo-r8

E por fim o reboot que inicializa o sistema instalado e configurado.

Após o reinício do sistema adicionamos um utilizador ao sistema:

```
useradd -m -G users -s /bin/bash irsuser

// criar a password para o user
passwd irsuser
```

Instalação de software

A instalação de software no gentoo é efectuada pela ferramenta emerge, interface do sistema de gestão de pacotes de software do Gentoo: Portage.

// instalação de pacote
emerge -av <pacote>

// actualização
emerge -uv <pacote>

// actualiza lista de pacotes
emerge –syn

// actualização global dos pacotes instalados
emerge -uDvaN world

Para a plataforma desenvolvida foram instalados diversos pacotes abaixo descriminados:

serviços	
	dhcpd
	apache
	bind
	quagga
	openvpn
	freeradius
outros serviços	
	syslog_mg
	vixio-cron
ferramentas	
	bind-tools
	wireshark
	mgrep
	vim
	pcutils
	gentoolkit
	grub
	iproute2

Configuração de serviços

Os serviços em Linux estão usualmente distribuídos por varios ficheiros, um ficheiro com o binário do serviço, um ou mais ficheiros de configuração do serviço, que utilizam uma semântica própria orientada á definição do comportamento do serviço em produção, e um script de gestão do serviço, distribuídos de forma predefinida pelo sistema de ficheiros:

/usr/sbin – Directoria com binários de sistema.

/etc – Directoria de ficheiros de configuração do sistema.

/etc/init.d - Directoria com os scripts de gestão de binários do sistema. Com parâmetros start, stop, restart e status.

Na tabela abaixo estão descriminados os serviços que implementamos, os ficheiros que os compõem bem como as directorias onde se encontram:

		ficheiros de configuração	**Script de gestão**	**serviço**
	Directoria base	**/etc/**	**/etc/init.d/**	**/usr/sbin/**
serviço				
apache		apache2/httpd.conf	apache2	apache2
ssh		ssh/sshd_config	sshd	sshd
quagga		quagga/zebra-conf	ospfd	ospfd
openvpn		openvpn/openvpn.conf	openvpn	openvpn
bind		bind/named.conf ……	named	named
freeradius		raddb/radiusd.conf …..	radiusd	radiusd

Apache

O apache implementa um servidor http, a configuração efectuada foi muito simples, apenas o suficiente para mostrar uma página de entrada.

No ficheiro de configuração */etc/apache2/httpd.conf* ajustamos a flag:

ServerRoot "/usr/lib/apache2"

Desta forma definimos a directoria raiz do servidor http, onde por omissão o browser espera encontrar o ficheiro index.html, que foi criado com uma mensagem de boas-vindas:

```
<html>
  <body>
    <h1>
      Welcome to IRS G3 homepage!
    </h1>
  </body>
</html>
```

Bind

O bind implementa um serviço de resolução de nomes, adicionamos á configuração de named.conf o nome atribuído á zona local definida.

```
zone "g3.lrcd.local" IN {
        type master;    // master of this domain
        file "pri/g3.lrcd.local.zone";  // descritor
        allow-update { none; };
        notify no;
};
```

Cria-se um ficheiro de configuração (*g3.lrcd.local.zone*) para a zona local criada, em */etc/bind/pri*:

```
$TTL 1W
;
;    Source of Authority
@      IN     SOA     ns.g3.lrcd.local. root.localhost.  (
                          2008122601 ; Serial
                          28800      ; Refresh
                          14400      ; Retry
                          604800     ; Expire - 1 week
                          86400 )    ; Minimum

;
;      Name Server
@            IN     NS     ns
;
;      Address
localhost       IN     A      127.0.0.1
ns              IN     A      10.62.74.200
azarujinha-irs  IN     A      10.62.74.200
;
;      aliases
www             IN     CNAME   ns
;
;
g3.lrcd.local   IN     A      10.62.74.200
```

Podemos verificar a resolução de nomes efectuada na máquina cliente:

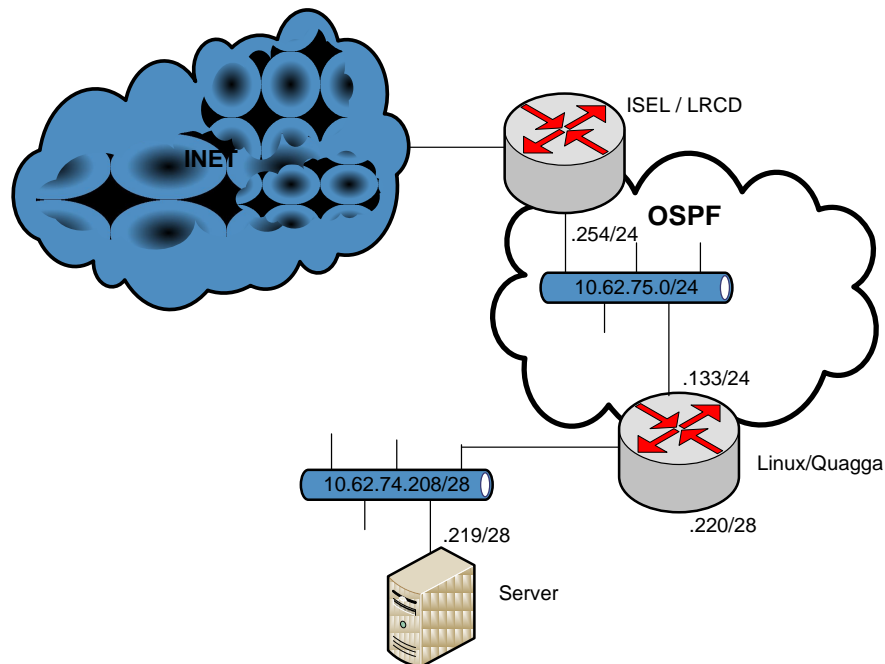
```
irsuser@azarujinha-irs-pc:~$ nslookup ns.g3.lrcd.local
Server:         10.62.74.220
Address:        10.62.74.220#53

Name: ns.g3.lrcd.local
Address: 10.62.74.200

irsuser@azarujinha-irs-pc:~$ nslookup www.g3.lrcd.local
Server:         10.62.74.220
Address:        10.62.74.220#53

www.g3.lrcd.local      canonical name = ns.g3.lrcd.local.
Name: ns.g3.lrcd.local
Address: 10.62.74.200
```

Quagga

O serviço quagga implementa funcionalidades de encaminhamento de tráfego IP, um router, por forma a demonstrar o seu funcionamento configurou-se o sistema para servir de interlucotor entre a rede local 10.62.74.208/28 e a rede do ISEL, trocando com esta rotas via protocolo de encaminhamento *OSPF*.

Por forma a que o sistema operativo possibilite o encaminhamento de trafego de rede entre diversas interfaces de rede alteramos a flag responsável por permitir o encaminhamento de pacotes IPv4, no ficheiro de configuração do kernel, as alterações têm efeito em tempo de execução (imediato):

vi /etc/sysctl.conf

 # enables packet forwarding
net.ipv4.ip_forward = 1

Na directoria */etc/quagga* podemos configurar cada protocolo de encaminhamento em ficheiros próprios: *zebra.conf, ospfd.conf*, etc.

Alternativamente á usual escrita nos ficheiros de configuração, o quagga disponibiliza a aplicação de configuração vtysh que nos fornece um interface de configuração á “ la cisco”, e é esta que se utilizou para a configuração do protocolo de encaminhamento estabelecido entre o nosso sistema e a rede do laboratório de rede. A configuração gerada via *vtysh* é guardada em *vtysh.conf*

```
azarujinha-irs# sh run
Building configuration...

Current configuration:
!
hostname azarujinha-rt
!
password irs0910
!
interface eth0
 ip ospf message-digest-key 20 md5 mesmosimples
 ipv6 nd suppress-ra
!
interface eth1
 ipv6 nd suppress-ra
!
interface lo
!
router ospf
 ospf router-id 10.62.74.208
 auto-cost reference-bandwidth 1000
 passive-interface eth1
 network 10.62.74.208/28 area 0.0.0.75
 network 10.62.75.0/24 area 0.0.0.75
 area 75 authentication message-digest
!
ip forwarding
!
line vty
!
```

Para verificar as tabelas de encaminhamento utilizamos os comandos de análise:

sh ip route ospf ou *debug*

```
azarujinha-irs# debug ospf
  event   OSPF event information
  ism     OSPF Interface State Machine
  lsa     OSPF Link State Advertisement
  nsm     OSPF Neighbor State Machine
  nssa    OSPF nssa information
  packet  OSPF packets
  zebra   OSPF Zebra information
```

FreeRadius

O serviço freeradius implementa um servidor de autenticação, acesso e contabilidade (*accounting*) Radius, contrariamente a outros como outros serviços a sua configuração está distribuída por vários ficheiros na directoria /etc/raddb, o radiusd.conf é o principal e congrega as configurações globais do serviço.

O ficheiro com configuração dos clientes que se pretendem autenticar no radius *clients.conf* é o que nos interessa abordar para tal criamos dois perfis de cliente, no sentido de suplicante que pretende ser autenticado. Para cada um definimos um alias e um segredo partilhado (entre o NAS e o servidor RADIUS).

```
client 10.62.74.208/28 {
      secret        = lanpasswd
      shortname      = lan
}

client localhost {
      ipaddr = 127.0.0.1
      shortname      = localhost
      secret        = radiuspasswd
}
```

Iniciamos o serviço em linha de comando para permitir uma melhor analise:

```
/usr/sbin/radiusd -X
```

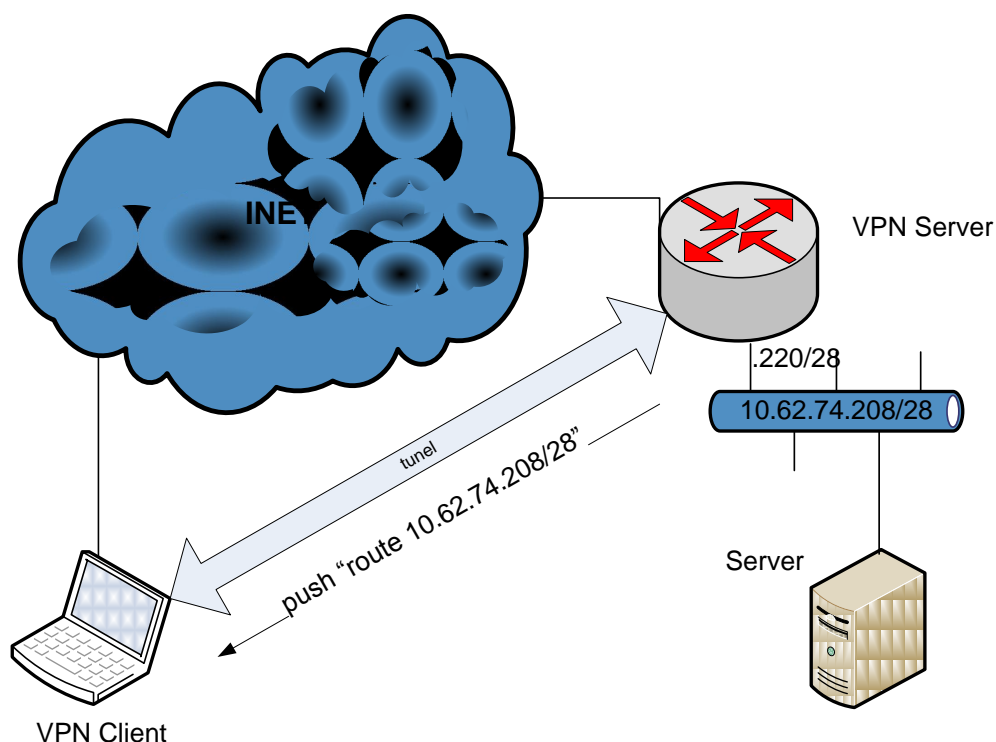
O serviço disponibiliza ainda um conjunto de ferramentas de operação, na directoria */usr/bin/* (começam por *rad**), nomeadamente a *radtest* ferramenta de teste do serviço.

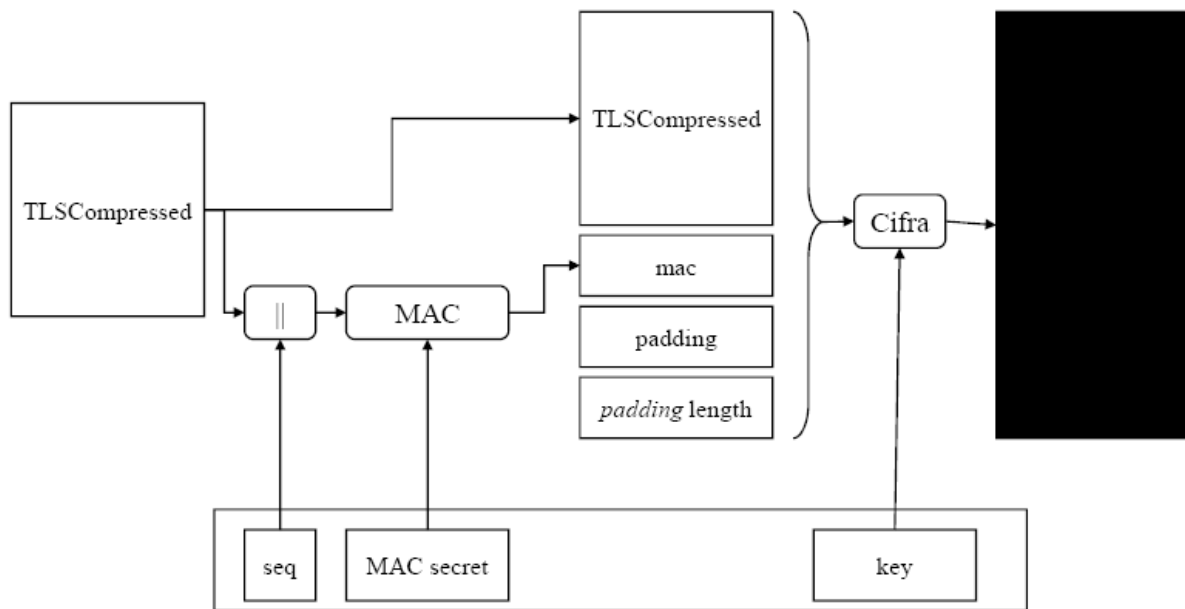
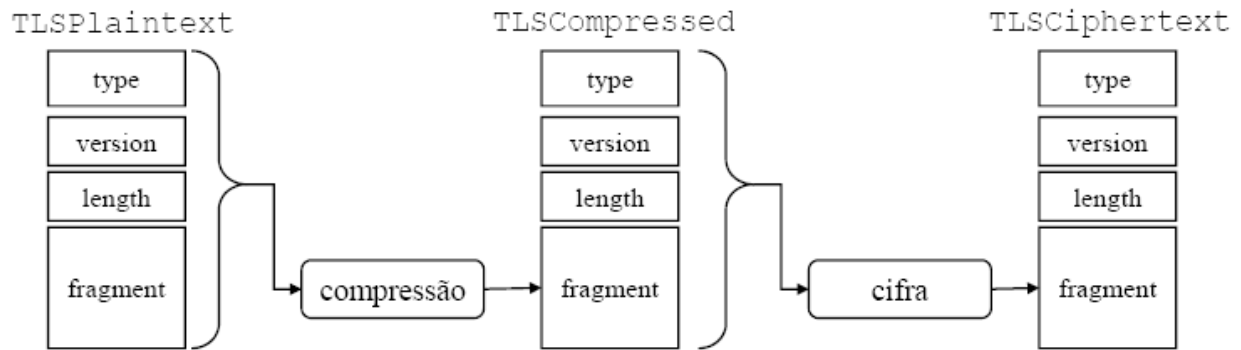
```
// teste sobre o localhost
azarujinha-irs bin # radtest irsuser irs0910 localhost 10 radiuspasswd
Sending Access-Request of id 31 to 127.0.0.1 port 1812
        User-Name = "irsuser"
        User-Password = "irs0910"
        NAS-IP-Address = 10.62.74.220
        NAS-Port = 10
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=31, length=20

// teste sobre a maquina azarujinha-irs (10.62.74.220)
azarujinha-irs bin # radtest irsuser irs0910 azarujinha-irs 10 lanpasswd
Sending Access-Request of id 55 to 10.62.74.220 port 1812
        User-Name = "irsuser"
        User-Password = "irs0910"
        NAS-IP-Address = 10.62.74.220
        NAS-Port = 10
rad_recv: Access-Accept packet from host 10.62.74.220 port 1812, id=55, length=20
```

OpenVpn

O openvpn implementa um serviço de VPN entre servidores e clientes, mais do que apenas o estabelecimento de túneis entre maquinas a VPN permite também a troca de parâmetros de rede (rotas, etc) entre as entidades que estabelecem o tunel, este visa garantir a troca de informação de forma confidencial, com integridade, autenticação mutua das entidades e não-repudiação.

Para a demonstração do serviço foi implementado o cenário mostrado na figura abaixo envolvendo um servidor, dois clientes (um Linux e um win 7), e uma maquina na LAN privada do servidor.

TLS

O TSL é um protocolo que permite a criação de túneis entre entidades, estes túneis têm como características a autenticação, integridade e não-repudiação através de certificados digitais, confidencialidade através de algoritmos de cifra, Blowfish, AES, etc. Permite ainda a compressão da informação antes da cifra. O TSL tem uma fase inicial de negociação em que são trocadas informações (cifras utilizadas, compressão (sim ou não), certificados digitais, etc) de forma a estabelecer o tunel.

O servidor vai produzir certificados digitais X509 e chaves, para o servidor e clientes assumindo o papel de autoridade certificadora (certification authority).

Copiamos as ferramentas para uma directoria de trabalho

cp -a /usr/share/openvpn/easy-rsa/ /etc/openvpn/

Gerar chaves RSA numa subdirectoria *keys* criada para o efeito.

Editar o vars e preencher os campos que identificam o *distinguished names*

export KEY_COUNTRY="PT"
export KEY_PROVINCE="LX"
export KEY_CITY="Lisboa"
export KEY_ORG="ISEL"
export KEY_EMAIL="19710@alunos.isel.ipl.pt"

Criar o certificado raiz (*certification authority*), utilizado por servidor e clientes:

azarujinha-irs easy-rsa # ./clean-all
azarujinha-irs easy-rsa # ./build-ca
Generating a 1024 bit RSA private key
....................................++++++
.................++++++
writing new private key to 'ca.key'

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [PT]:
State or Province Name (full name) [LX]:
Locality Name (eg, city) [Lisboa]:
Organization Name (eg, company) [ISEL]:
Organizational Unit Name (eg, section) []:IRS
Common Name (eg, your name or your server's hostname) [ISEL CA]:azarujinha-irs
Name []:azarujinha-irs
Email Address [19710@alunos.isel.ipl.pt]:
azarujinha-irs easy-rsa #

Assim são criados os certificados na subdirectoria keys, nomeadamente o certificado raiz (*certification authority*) *ca.crt*. Esta não é uma situação invulgar, usualmente os certificados já existem nos sistemas operativos, providenciados pelos fornecedores de software, mas um certificado assinado por uma entidade "confiável" tem custos. Assim para a nossa VPN o openvpn disponibiliza os mecanismos para criarmos os nossos certificados.

A criação de certificados por cliente (funcionário, aluno, etc) visa garantir a autenticação e não-repudiação da comunicação.

./build-key-server servidor
./build-key cliente
./build-key cliente1
…..

Geração dos parâmetros diffie-helman *dh1024.pem* para a distribuição segura de chaves entre cliente e servidor.

./build-dh

Criar a directoria para guardar as chaves e guardar lá os ficheiros necessários

mkdir /etc/openvpn/keys
cp -a ca.crt servidor.crt servidor.key dh1024.pem /etc/openvpn/keys/

Agora há que passar os certificados ao cliente (uma vez que aqui não temos um método automático de distribuição tem de ser feito manualmente (dentro do espírito da cadeira)).

Cria-se a directoria keys e copiam-se os ficheiros relevantes para o cliente.

sftp 192.168.1.72

```
sftp>cd /etc/openvpn/easy-rsa/keys
sftp> put ca.crt
sftp> put dh1024.pem
sftp> put cliente.key
sftp> put cliente.crt
```

O openvpn permite que o tunel *TLS* transporte pacotes IP ou tramas *Ethernet*, para isso a ligação assenta em uma de duas interfaces lógicas: *tun IP* ou *tap Ethernet*, na demonstração utilizou-se IP sobre IP, pois no cenário pretendido não se verificou vantagem em transportar Ethernet, face ao over-head que resulta de ter de se enviar também o cabeçalho *ethernet*.

O ficheiro de configuração é muito sucinto e objectivo, cada linha define um parâmetro, de entre os que foram abordados anteriormente, e é muito idêntico entre distribuições do openvpn para os vários sistemas operativos. O ficheiro esta devidamente comentado.

Configuração do servidor

```
#
# Config do servidor
#
# device do tunel TUN/TAP
# tun - IP sobre IP
# tap - Eth sobre IP
dev tun
#
# dir de configs
cd /etc/openvpn
#
# Autenticacao, integridade, nao-repudiacao
tls-server
#
# certificado raiz e do servidor
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/servidor.crt
#
# chave do servidor
key /etc/openvpn/keys/servidor.key
#
# Diffie-Hellman
dh /etc/openvpn/keys/dh1024.pem
```

```
#
# cifra
#cipher BF-CBC
cipher AES-256-CBC
#
# compressao
comp-lzo
#
# port UDP 1194
proto udp
port 1194
#
# enderecamento
# pool 10.1.2.0/24
# servidor 10.1.2.1/24
# clientes 10.1.2.[2-254]/24
server 10.1.2.0 255.255.255.0
#
# push - directivas enviadas para os clientes
# rotas
push "route 10.0.0.0 255.255.255.0"
push "route 10.62.74.208 255.255.255.240"
#
# user e grupo OpenVPN (nao se usa no WIN)
user nobody
group nobody
#
# keepalive da conexao
# pings cada 10s
# mantem conexao ate 120s sem tráfego
keepalive 10 120
#
# logs and status
;status openvpn-status.log
;log openvpn.log
;log-append openvpn.log
verb 3
#
# END
```

A configuração do cliente

```
#
# config do cliente
#
# device do tunel TUN/TAP
# tun - IP sobre IP
# tap - Eth sobre IP
dev tun
```

```
#
# dir do openvpn
cd /etc/openvpn
#
# Autenticacao, integridade, nao-repudiacao
tls-client
#
# certificado raiz e do servidor
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/cliente.crt
#
# chave do cliente
key /etc/openvpn/keys/cliente.key
#
# Diffie-Hellman
dh /keys/dh1024.pem
#
# cifra
cipher AES-256-CBC
#
# compressao
comp-lzo
#
# UDP port 1194
proto udp
port 1194
#
# enderecamento
# dado pelo servidor
client
#
# IP (publico) do servidor
remote 192.168.1.64
#
# user/group OpenVPN (nao se usa no WIN)
user nobody
#group nobody
group nogroup
#
# keepalive.
keepalive 10 120
#
# logs and status
#status openvpn-status.log
#log openvpn.log
#log-append openvpn.log
verb 3
```

Para uma melhor analise corremos o servidor em comando de linha (com os logs comentados):

/usr/sbin/openvpn --config /etc/openvpn/openvpn.conf

Com a instancia do serviço é criado um interface de tunel

```
ifconfig
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.1.2.1  P-t-P:10.1.2.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 B)  TX bytes:32 (32.0 B)
```

Corremos o cliente também em comando de linha:

/usr/sbin/openvpn --config /etc/openvpn/irs-vpn-client.conf

Tambem aqui é criado uma interface lógica *tun0*

```
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.1.2.6  P-t-P:10.1.2.5  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Uma analise das rotas:

```
root@azarujinha-1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.2.5        *               255.255.255.255 UH    0      0        0 tun0
10.1.2.1        10.1.2.5        255.255.255.255 UGH   0      0        0 tun0
10.62.74.208    10.1.2.5        255.255.255.240 UG    0      0        0 tun0
10.0.0.0        10.1.2.5        255.255.255.0   UG    0      0        0 tun0
192.168.1.0     *               255.255.255.0   U     1      0        0 eth0
192.168.250.0   *               255.255.255.0   U     1      0        0 eth1
link-local      *               255.255.0.0     U     1000   0        0 eth1
default         home            0.0.0.0         UG    0      0        0 eth0
```

Através destas observações verificamos a seguinte configuração de rede implementada pelo serviço:

A interface criada é a 10.1.2.6/30 com next-op o .5, ao nível das rotas verificamos que o .5 está encaminhado para o tunel, tal como o .1 (IP do servidor), bem como as rotas de LAN privada 10.62.74.208/28 recebidas do servidor. Com a ligação de mais clientes verificou-se que por cada é criada uma ligação /30 entre cada um e o servidor (10.1.2.6, .10, etc).

A partir do remoto efectuamos um teste de conectividade á LAN privada do servidor:

root@azarujinha-1:~# ping 10.62.74.220
PING 10.62.74.220 (10.62.74.220) 56(84) bytes of data.
64 bytes from 10.62.74.220: icmp_seq=1 ttl=64 time=3.83 ms

O cliente windows do openvpn oferece um ambiente gráfico onde se pode observar a evolução do processo de ligação ao servidor, bem como a interface criada para suportar o tunel de ligação ao servidor e as rotas recebidas:

Conclusão

O sistema implementado permitiu o desenvolvimento de uma solução de integração de serviços de rede, assente numa plataforma open-source, e a respectiva implementação em situações reais.

A instalação e configuração do sistema operativo permitiu uma abordagem prática a aspectos como sistemas de ficheiros, instalação de pacotes de software, activação de serviços, drivers, interfaces, etc. Fazendo uso dos conhecimentos adquiridos ao nível de sistemas operativos e arquitectura de computadores, o desenvolvimento passou tambem pela instalação, configuração e teste de serviços de rede, tendo para o efeito baseado o trabalho nas competencias adquiridos, essencialmente, nas disciplinas de redes, bem como em documentação disponibilizada na internet por fornecedores de software, instituições de ensino, foruns tematicos de Linux, etc.

Referencias

Links

http://www.gentoo.org/

http://www.vivaolinux.com.br

http://www.linfo.org/index.html

http://www.guiadohardware.net

http://openvpn.net/index.php/open-source/documentation/howto.html

Documentos

http://i.techrepublic.com.com/downloads/PDF/SolutionBase_RADIUS_deployment_scenarios.pdf

http://www.sans.org/reading_room/whitepapers/vpns/openvpn_and_the_ssl_vpn_revolution_1459